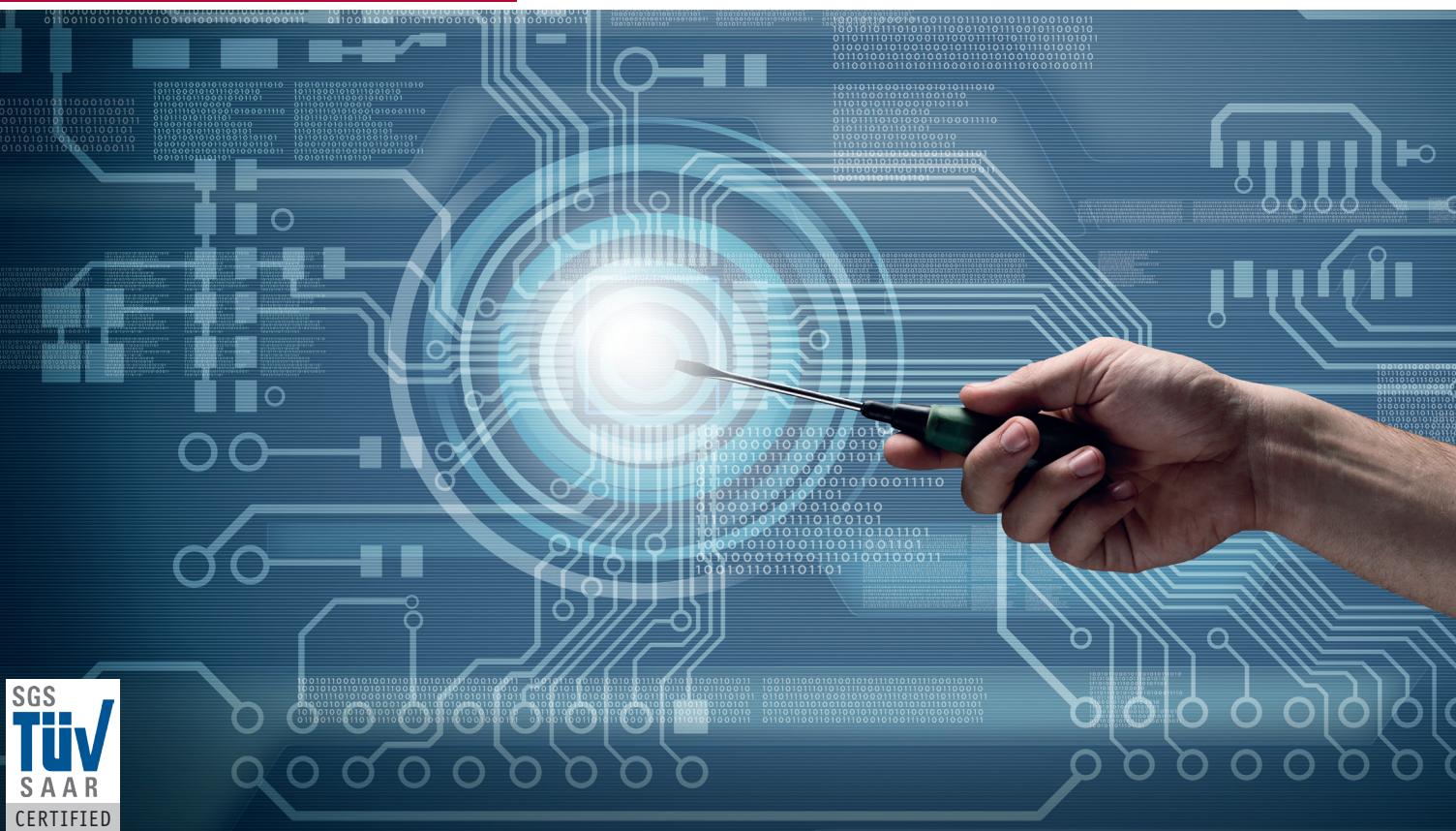


Dynamic Testing

CANTATA



Accelerate safety standards compliance
for C & C++ with Cantata automated
unit & integration testing

QA SYSTEMS

The Software Quality Company

Driving embedded
software quality



Why Industry leaders use Cantata

Cut the cost of standards compliance

Cantata meets the dynamic testing requirements of software safety standards. It is a single solution for dynamic unit and integration testing on host and target platforms.

Certification of development tools can be a heavy compliance cost burden. Cantata has been independently certified by SGS-TÜV SAAR GmbH, and provides a tool certification kit with everything needed out-of-the-box, available free of charge.

Understanding how to comply with software safety standards is complex and time consuming. Comprehensive guidance is provided free of charge for using the full feature set in Cantata to meet dynamic testing requirements of specific standards. This combination of guidance and Cantata capabilities accelerates the achievement of standards compliance.



TÜV certified for:

- › ISO 26262:2011 (Automotive)
- › EN 50128:2011 (Railways)
- › IEC 62304:2006 (Medical)
- › IEC 61508:2010 (General Industrial)
- › IEC 60880:2006 (Nuclear Power)

Qualifiable for:

- › DO-178B
- › DO-178C / DO-330
- › Other standards as required

Reduce risk of software failure

Product recalls and infection of the wider brand and corporate reputations can far exceed the development cost of each application. Unit testing is the most thorough way to test application code and prevent bugs within shipping devices.

Project over-runs can be mitigated by shifting verification effort to the earliest stages in the software development lifecycle. This reduces the risks of delays during later testing stages because unit tested components are easier and more predictable to integrate.

Fitness for purpose litigation against companies and individuals is now an increasing risk. Where companies fail to employ accepted industry practices such as thorough unit testing with Cantata, they cannot use the “state of the art” legal defence against such litigation.

Lower testing costs

Testing earlier lowers costs by minimising code re-work later in the development lifecycle. Developers can identify defects with Cantata unit and integration testing as soon as each component is available.

The high cost of standards compliant unit and integration testing can be dramatically lowered through automation. Satisfying the dynamic testing requirements of safety standards is accelerated by Cantata automation of:

- | | |
|---------------------------------|--|
| › Test framework generation | › Test execution on host or target |
| › Test case generation | › Regression testing in Continuous Integration |
| › Baseline tests on legacy code | › Results diagnostics and report generation |

Integrating tools into a toolchain can add hidden testing costs. Cantata’s tight integration with cross-compilation environments, DevOps workflows and its intuitive C/C++ code tests in Eclipse® GUI or code editors make it easy to slot into any toolchain. These integrations lower the tool learning curve and accelerate the testing activity, lowering overall testing costs.

Shorten time to market

Industry leaders recognise the need to ship faster without endangering quality. Cantata tests provide two key time advantages for development managers:

- › Team collaboration and efficiency is improved with structured consistent tests and certification ready reports.
- › Integration times are shorter and more predictable when integrating individually proven software components.

Cantata works with your environment

Cantata highlights

- › Automated test harness and test case creation.
- › Extensive platform support and toolchain integrations.
- › Easy to use Eclipse® GUI and tests written in C/C++.
- › Bi-directional requirements traceability.
- › Unique call interface control to simulate and intercept calls.
- › Flexible user code injection.
- › Support for Test Driven Development
- › Integrated code coverage analysis.
- › Automated regression testing.
- › Test maintenance for code changes.
- › Free tool certification kit for all major safety standards.

Cantata installs on Windows® and Linux® host operating systems, with a Built-on-Eclipse® IDE or as an Eclipse-Ready® plug-in set. It supports GCC and Microsoft Developer Studio® compilers, and is also integrated with an extensive set of embedded development toolchains:

- | | |
|--|--|
| <input checked="" type="checkbox"/> IDEs / RTOSs
<input checked="" type="checkbox"/> Cross-compilers
<input checked="" type="checkbox"/> Debuggers | <input checked="" type="checkbox"/> Build / Continuous Integration
<input checked="" type="checkbox"/> Software Configuration Management
<input checked="" type="checkbox"/> Requirements Management |
|--|--|

To confirm how your tools and platforms are supported, please contact QA Systems.

Unrestricted embedded target use

For target environments, a built-in wizard provides unlimited deployment and use without any license restrictions. Cantata deployments consist of libraries and configuration options, binary compatible with your code. These are tested and controlled for certified use on safety related projects. Deployment switching makes it easy to use one set of tests across multiple targets for different product variants.

Easy and flexible testing on target

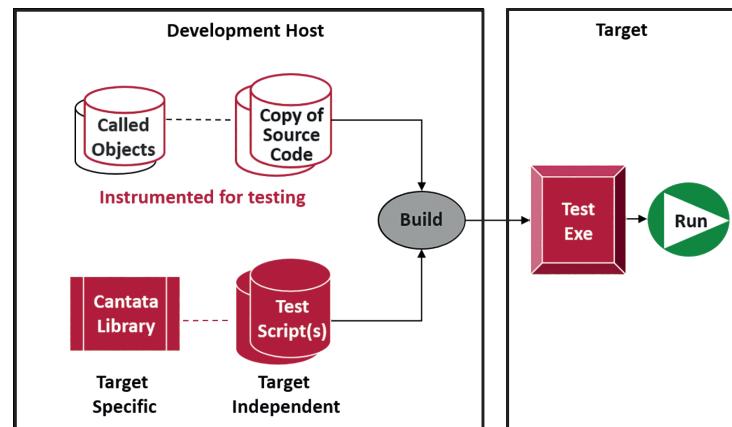
Cantata tests (platform independent test scripts in C/C++ and platform specific deployments) are built as C/C++ executables, downloaded and run on a target platform just as you would with your own code on a:

- Simulator Emulator Physical Target Board

Functional and code coverage test results are directed back to the host for diagnostics and reporting. The process is completely automated using Cantata Makefiles, test scripts and platform customisations for easy and flexible on-target testing from the GUI or CLI.

Types of testing supported:

- black-box / white-box
- positive / negative
- requirements / robustness
- single / large input data sets
- procedural / object oriented
- from code / from headers
- call simulation / interception
- isolation / integration
- host / target execution
- new / regression



User code is driven by portable test scripts with target compatible libraries, and built as a single test executable to run on multiple host or target platforms. Instrumentation is used for white-box access and code coverage, so production code is never modified for testing.

Flexible test framework

A flexible test framework (test scripts and supporting library) enables any combination of testing styles for both unit and scalable integration testing. Tests can be edited in a GUI or directly as C/C++ code, and run as executables on both embedded target and host platforms.

Test Driven Development (TDD)

Cantata allows tests to be written as soon as function prototypes are created within header files. This allows the test framework to be constructed before the body of the software under test is fully implemented. Cantata for TDD enhances traditional black-box TDD techniques, by giving access to fully featured white-box testing on encapsulated code internals such as private/static data and functions.

Black and white box testing

Highly automated test cases provide power and precision for black-box testing, and more efficient thorough white-box testing. Black-box testing is enabled with user-selected or pre-defined parameterised looping tests, a combinatorial effect calculator, and CSV import/export for large data sets.

Precise white-box testing via Cantata instrumentation automatically accesses encapsulated code directly from the test script, without conditional compilation, giving control over both static and private functions and data. Users can inject extra code (via instrumentation) for testing purposes, without modifying the production code.

Robustness testing

Robustness testing is made easy with Cantata Rule Sets of pre-defined values for basic data types, in looping test cases. All accessible global data is automatically checked for inadvertent changes.

Object oriented testing

Cantata object oriented style tests are implemented as classes for testing methods, templates, or classes. They feature automated:

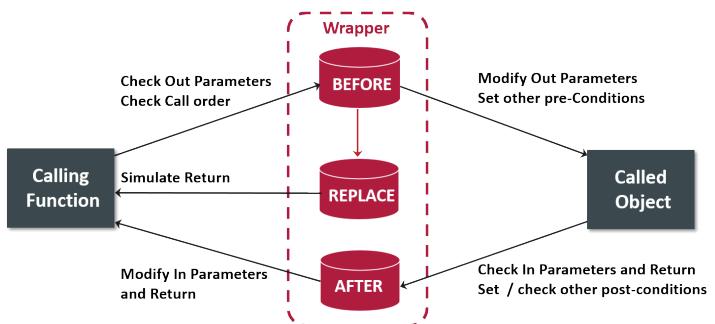
- › test case re-use via a parallel inheritance hierarchy
- › test class inheritance structure for inherited classes
- › concrete implementation of abstract base classes (ABCs) or pure virtual methods (PVMs).
- › resolution of dependencies on undefined references that are not directly called by the code.

Unique call control

Cantata automatically generates test controls to both simulate (stub) and intercept (wrap) all function calls from the software under test, providing:

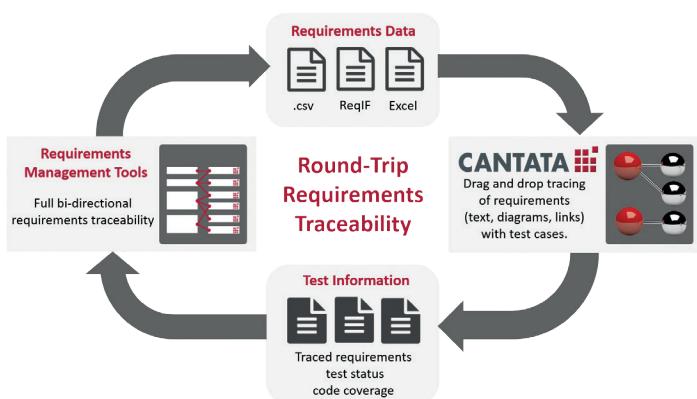
- › optional automatic checks on parameters and data
- › multiple instances for differential call behaviour
- › flexible call order verification in each test case
- › interface error detection and error injection
- › control coupling testing

Wrappers intercept calls to verify the actual, not the assumed, or simulated behaviour of the called object. Where simulation is not possible or desirable (internal calls at integration, OS calls, hardware interfaces etc.), wrappers provide powerful call control unique to Cantata.



Requirements tracing

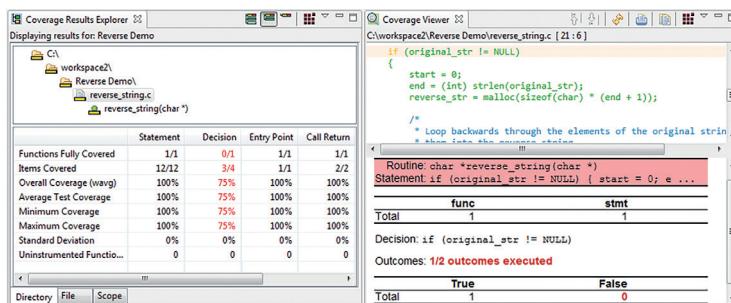
Requirements or test plan sets are imported to a Cantata server. Relationships are assigned with a drag-and-drop interface, and then exported with Cantata test results status and code coverage information for bi-directional traceability.



Import / export can be via CSV, Microsoft Excel®, or requirements interchange format (ReqIF), to fit your workflow and specific requirements management tool version (e.g. IBM® Rational® DOORS®, PTC Integrity®, Polarion® REQUIREMENTS™, Intland codeBeamer®).

Code coverage

Cantata code coverage provides objective measurement of how thoroughly tests have executed the source code (whether or not driven by Cantata tests). Standard specific Cantata coverage Rule Sets make it easy to use by automating the instrumentation, data reporting and integrated checking of required code coverage levels.



Code Coverage is measured using the following metrics:

- › Entry points
- › Conditions
- › Call Returns
- › MC/DC
- › Statements
- › Loops
- › Basic Blocks
- › Relational Operators
- › Decisions (branches)

Cantata Build Variant Coverage gathers data on source code, when defines are used to build executable code in different variants, providing analysis and certified reporting of coverage aggregated over all variants.

Pin-point diagnostics can filter or aggregate coverage for complete project code-trees, drilling down to individual code constructs, within each line of code, by test case, test run and metric type and code context (inheritance, threads, states, data coupling etc) and source build variant. Automatic test case vector selection from large data sets, and reduces regression testing overhead.

Continuous testing



Cantata Makefiles are automatically generated at test creation to compile, link, deploy, execute and retrieve results for suites of tests in batch mode. They can be used with existing makefiles, and are easily integrated with open source or commercial continuous integration tools.

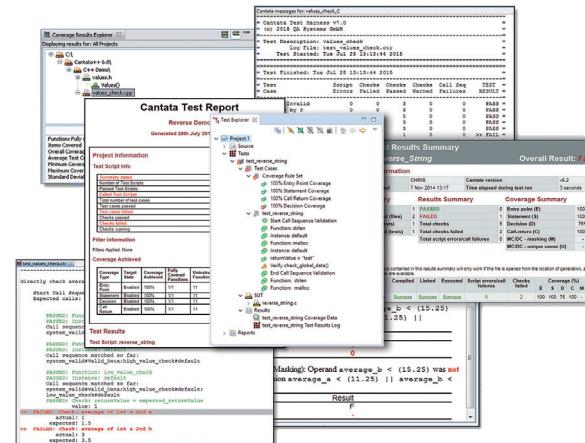
AutoTest

Test case vectors can exercise 100% code coverage while checking data, parameters and call order. AutoTest creates a safety net of regression unit tests to reduce reliance on expensive system tests, and automatically closes gaps in code coverage.

Code Change Analysis

Cantata Code Change Analysis helps automate unit test maintenance. Changes which impact existing tests are identified and suitable updates suggested. Test scripts are then automatically refactored.

Diagnostics and reports



Cantata provides powerful filterable diagnostics of test and code coverage results within the Eclipse® GUI, and flexible user configurable reports in XML, HTML and certification-ready ASCII text.

Team Reporting - manager add-on

Cantata Team Reporting, with a client-server architecture, web interface and REST API, provides current testing status and historical trends over multiple codebases. See the Team Reporting datasheet for more details.



Customer testimonials

"We were highly impressed with Cantata's track record in testing high integrity software in avionics, military applications and, of course, in medicine. It was the natural choice for us." **Urs Reidt, Research and Development Director**



SAAB

"I've used Cantata for several years now to validate and test airborne safety-critical software. The tool has been great and quite simple in its full complexity."
"Cantata is easy to learn and use."

Johnny Johansson, Validation & Verification Tools Manager



"The systematic use of Cantata has enabled us to have the shortest unit test phase possible with great efficiency in terms of cost."

Philippe Lomazzi, Head of Software Development

TATA ELXSI

"Through years of experience in unit testing using Cantata for various platforms and languages... we are confident in suggesting Cantata to our clients."

Padmakumar TV, Senior Engineering Specialist



"Speaking conservatively, this product has probably paid for itself twice over already!"

John Duckett, Special Projects Manager

Bosch Engineering GmbH

"Module testing early during development is becoming more important due to the increasing complexity of software. In practice Cantata has proved successful and increased unit testing efficiency. Cantata from QA Systems offers an ideal solution for the creation and execution of unit and integration tests, including coverage analysis."

Matthias Schmidt, Testmanagement, Verification and Validation



Get a demo

Contact us to arrange a bespoke web or on-site demo of Cantata for your team's testing requirements.



Start free trial

Take Cantata for a test drive with your own code and development environment. Try the full tool with free technical support during your evaluation.



Learn more

Visit the Cantata website for more information.

www.qa-systems.com/tools/cantata